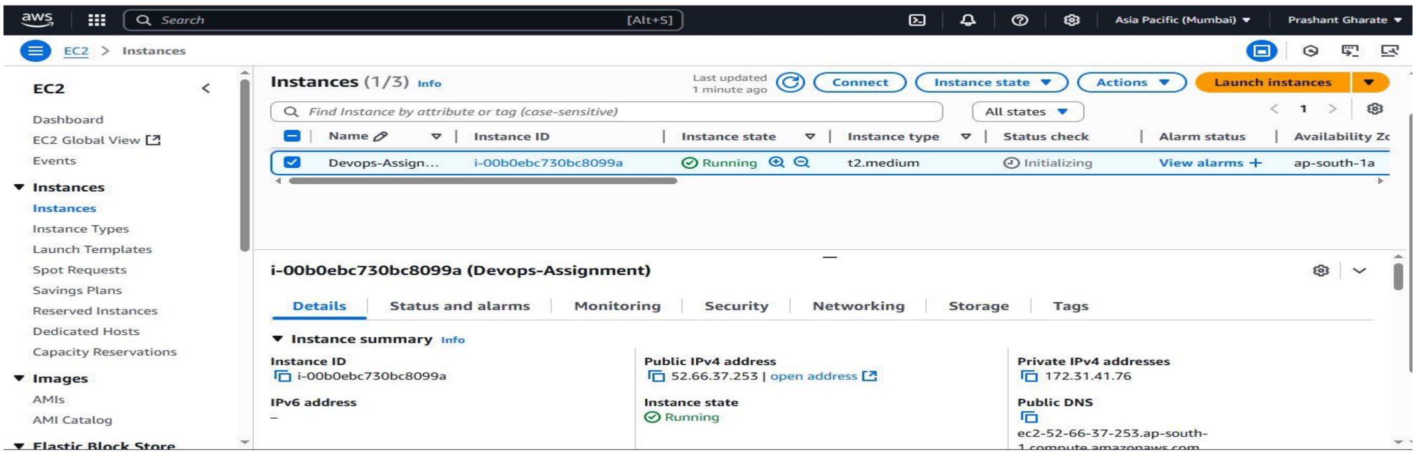
📄 Project Documentation: Jenkins Deployment with Domain and SSL using DuckDNS

1. Project Overview

This project demonstrates a complete CI/CD pipeline implementation for a Node.js web application using Jenkins, GitHub, PM2, NGINX, HTTPS (via Certbot), and AWS CloudWatch for monitoring. It includes Build → Test → Deploy automation with secure access and process monitoring.

◆ Step 1: Create an EC2 Instance



Instances (1/3) Info — Last updated 1 minute ago

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Devops-Assign...	i-00b0ebc730bc8099a	Running	t2.medium	Initializing	View alarms +	ap-south-1a

i-00b0ebc730bc8099a (Devops-Assignment)

Instance summary Info

Instance ID: i-00b0ebc730bc8099a

IPv6 address: –

Public IPv4 address: 52.66.37.253 | open address

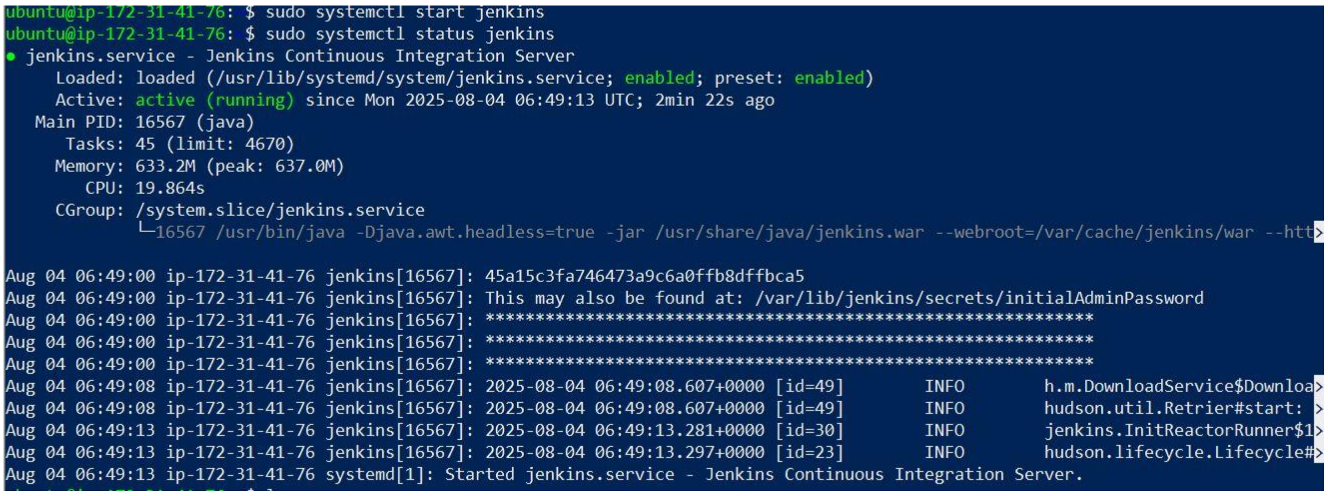
Instance state: Running

Private IPv4 addresses: 172.31.41.76

Public DNS: ec2-52-66-37-253.ap-south-1.compute.amazonaws.com

- Launch a new EC2 instance using Ubuntu (20.04 or later).
- Instance type: `t2.medium`.
- Name: `Devops-Assignment`.
- Region: `ap-south-1a` (Mumbai).
- Ensure that the instance is in a **Running** state.
- Note the **Public IPv4 address** (used for browser access).
- Example: `52.66.37.253`.

◆ Step 2: Install Jenkins and Start the Service



```
ubuntu@ip-172-31-41-76:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-41-76:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-08-04 06:49:13 UTC; 2min 22s ago
   Main PID: 16567 (java)
      Tasks: 45 (limit: 4670)
     Memory: 633.2M (peak: 637.0M)
        CPU: 19.864s
     CGroup: /system.slice/jenkins.service
             └─16567 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --htt>

Aug 04 06:49:00 ip-172-31-41-76 jenkins[16567]: 45a15c3fa746473a9c6a0ffb8dffbca5
Aug 04 06:49:00 ip-172-31-41-76 jenkins[16567]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 04 06:49:00 ip-172-31-41-76 jenkins[16567]: *************************************************************
Aug 04 06:49:00 ip-172-31-41-76 jenkins[16567]: *************************************************************
Aug 04 06:49:00 ip-172-31-41-76 jenkins[16567]: *************************************************************
Aug 04 06:49:08 ip-172-31-41-76 jenkins[16567]: 2025-08-04 06:49:08.607+0000 [id=49]        INFO        h.m.DownloadService$Downloa>
Aug 04 06:49:08 ip-172-31-41-76 jenkins[16567]: 2025-08-04 06:49:08.607+0000 [id=49]        INFO        hudson.util.Retrier#start: >
Aug 04 06:49:13 ip-172-31-41-76 jenkins[16567]: 2025-08-04 06:49:13.281+0000 [id=30]        INFO        jenkins.InitReactorRunner$1>
Aug 04 06:49:13 ip-172-31-41-76 jenkins[16567]: 2025-08-04 06:49:13.297+0000 [id=23]        INFO        hudson.lifecycle.Lifecycle#>
Aug 04 06:49:13 ip-172-31-41-76 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
```

SSH into your EC2 and run the following commands:

```
sudo apt update
sudo apt install openjdk-11-jdk -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins
sudo systemctl status jenkins
```

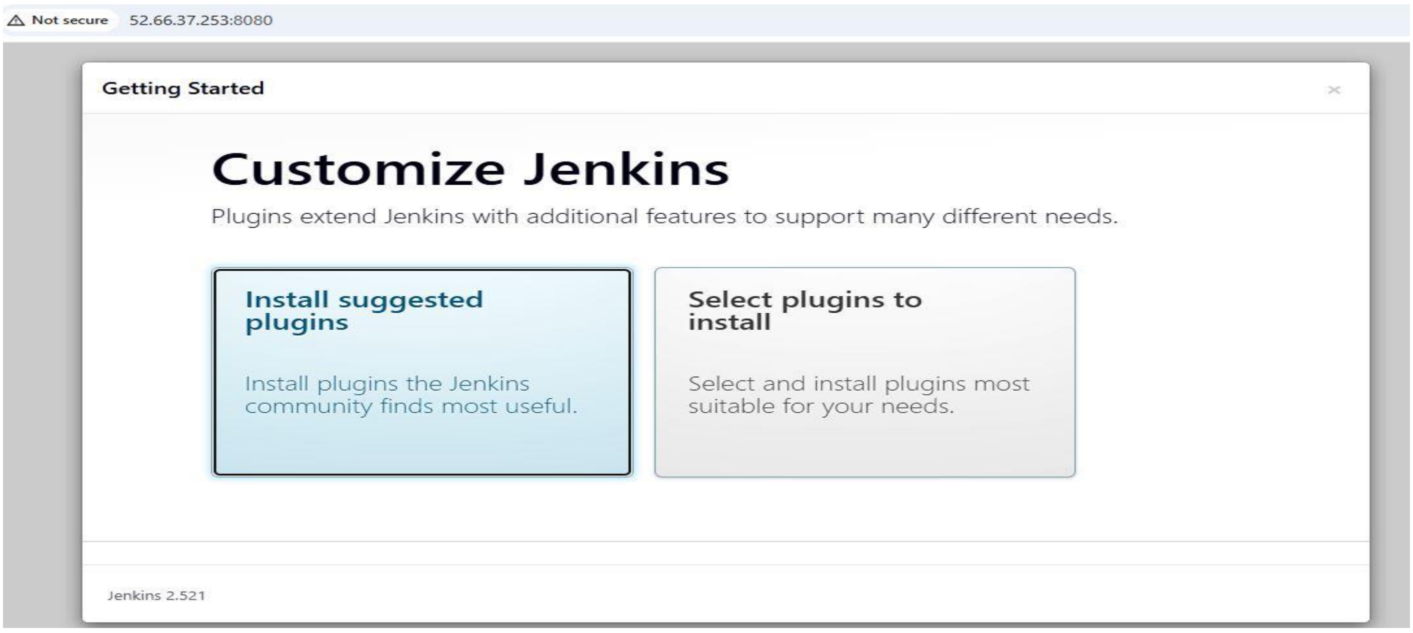
✓ You should see `Active: active (running)` for Jenkins service.

---

2. Tools & Services Used

- Jenkins (Automation Server)
- GitHub (Source Code Repository)
- Node.js (Application Platform)
- PM2 (Process Manager for Node.js)
- NGINX (Web Server & Reverse Proxy)
- Certbot + DuckDNS (HTTPS Configuration)
- AWS EC2 (Hosting)
- AWS CloudWatch (Monitoring)

Step 3: Jenkins Initial Setup


Not secure 52.66.37.253:8080

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

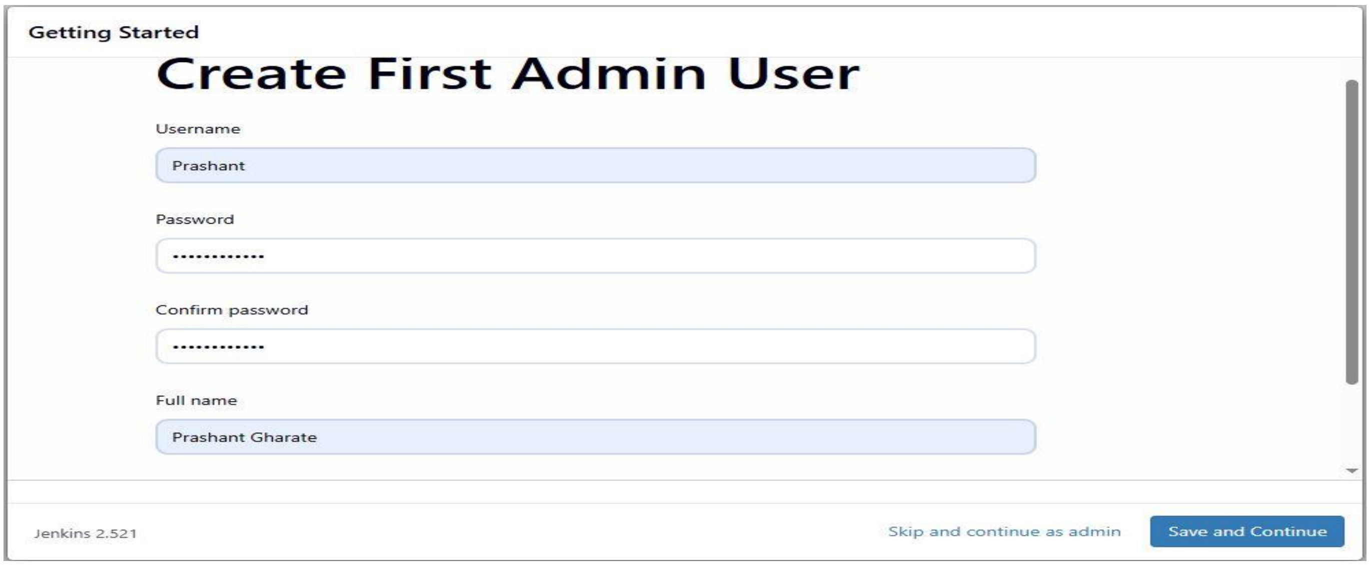
Jenkins 2.521

- Open Jenkins in browser:
  ☞ `http://<your-ec2-public-ip>:8080`
- You’ll be prompted to unlock Jenkins using a password found at:

```
bash
CopyEdit
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Paste the password in the browser and proceed.
- Select **Install Suggested Plugins** for simplicity.

Step 4: Create Jenkins Admin User



Create First Admin User

Username: Prashant

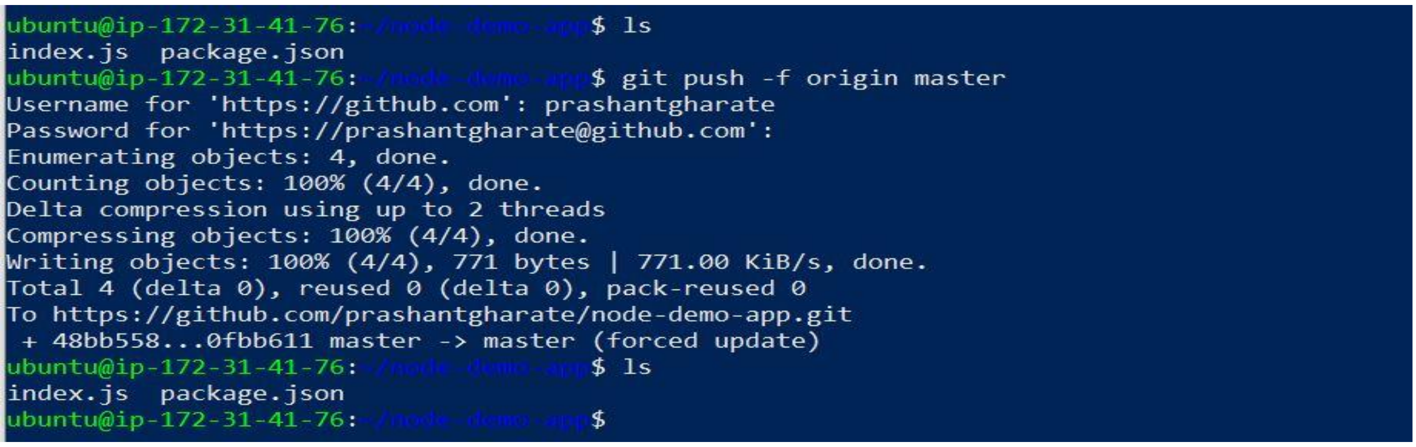
Password: ••••••••••••

Confirm password: ••••••••••••

Full name: Prashant Gharate

- Fill out the form with:
    - Username: `Prashant`
    - Password: `your-password`
    - Full Name: `Prashant Gharate`
- Click on **Save and Continue** to complete setup.

Step 5: Push Node.js App to GitHub



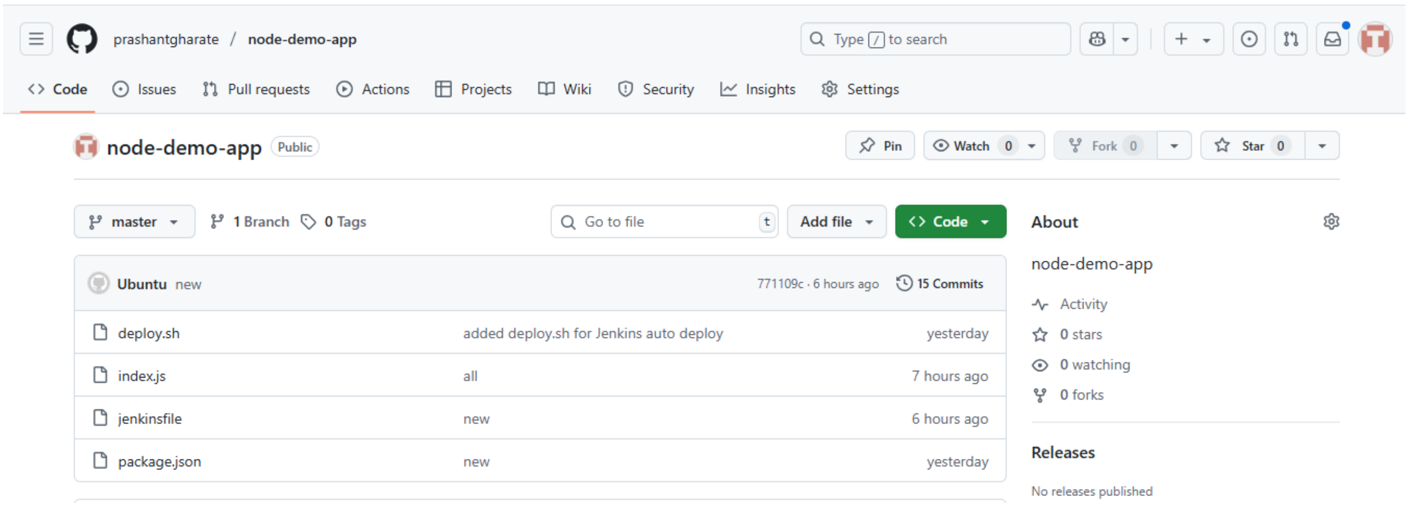
```
ubuntu@ip-172-31-41-76:~/node-demo-app$ ls
index.js  package.json
ubuntu@ip-172-31-41-76:~/node-demo-app$ git push -f origin master
Username for 'https://github.com': prashantgharate
Password for 'https://prashantgharate@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 771 bytes | 771.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/prashantgharate/node-demo-app.git
 + 48bb558...0fbb611 master -> master (forced update)
ubuntu@ip-172-31-41-76:~/node-demo-app$ ls
index.js  package.json
ubuntu@ip-172-31-41-76:~/node-demo-app$
```

- Create a directory `node-demo-app` and push files:

```
git init
git remote add origin https://github.com/prashantgharate/node-demo-app.git
git add .
git commit -m "Initial commit"
git push -f origin master
```

💼 Files pushed:

- `index.js`
- `package.json`
- `deploy.sh`
- `jenkinsfile`



prashantgharate / node-demo-app

node-demo-app Public

master · 1 Branch · 0 Tags

Ubuntu new · 771109c · 6 hours ago · 15 Commits

deploy.sh	added deploy.sh for Jenkins auto deploy	yesterday
index.js	all	7 hours ago
jenkinsfile	new	6 hours ago
package.json	new	yesterday

About

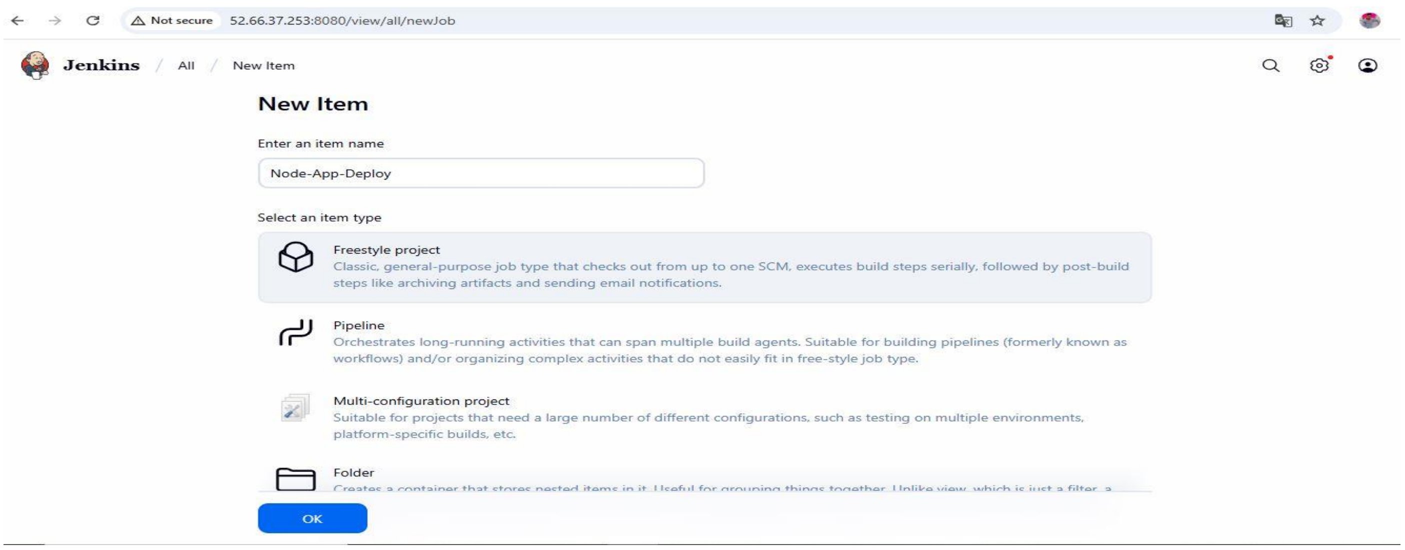
node-demo-app

Activity
0 stars
0 watching
0 forks

Releases

No releases published

Step 6: Create a Jenkins Job



Not secure 52.66.37.253:8080/view/all/newJob

Jenkins / All / New Item

New Item

Enter an item name

Node-App-Deploy

Select an item type

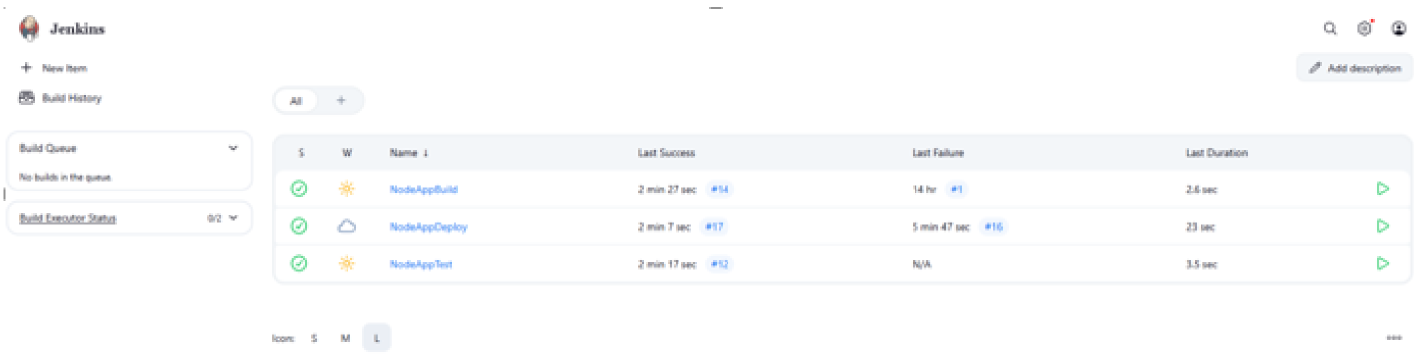
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

- Go to Jenkins Dashboard → **New Item**
- Enter name: `Node-App-Deploy`
- Choose: **Freestyle project**
- Click **OK**



Jenkins

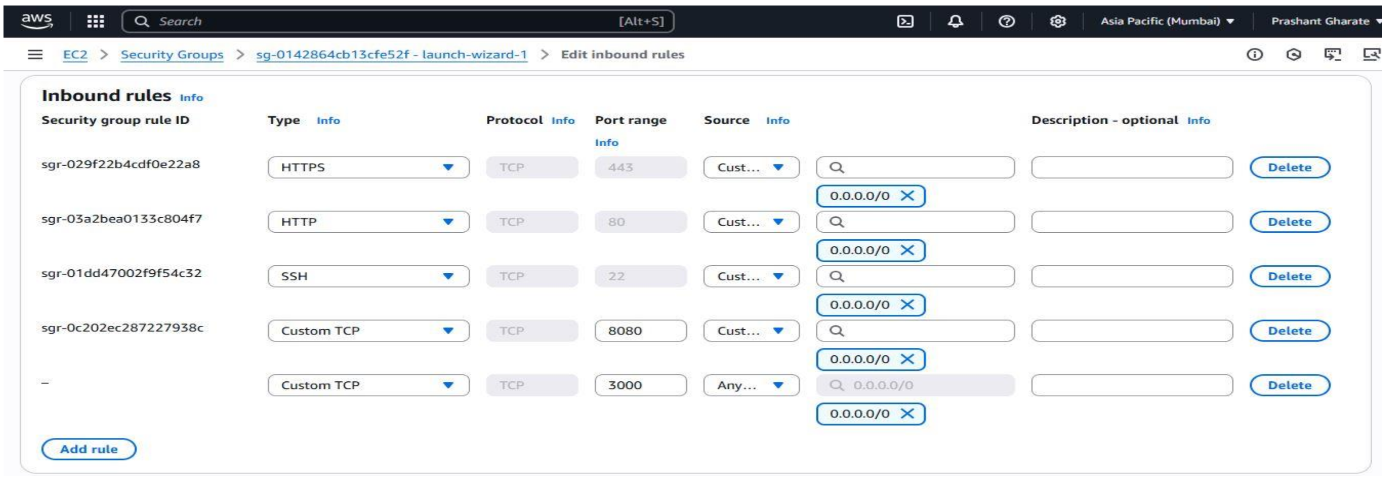
Build Queue
No builds in the queue.

Build Executor Status 0/2

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	NodeAppBuild	2 min 27 sec #14	14 hr #1	2.6 sec
✓	☁	NodeAppDeploy	2 min 7 sec #17	5 min 47 sec #16	23 sec
✓	☀	NodeAppTest	2 min 17 sec #12	N/A	3.5 sec

This will create a basic job for deployment.

Step 7: Configure Inbound Rules

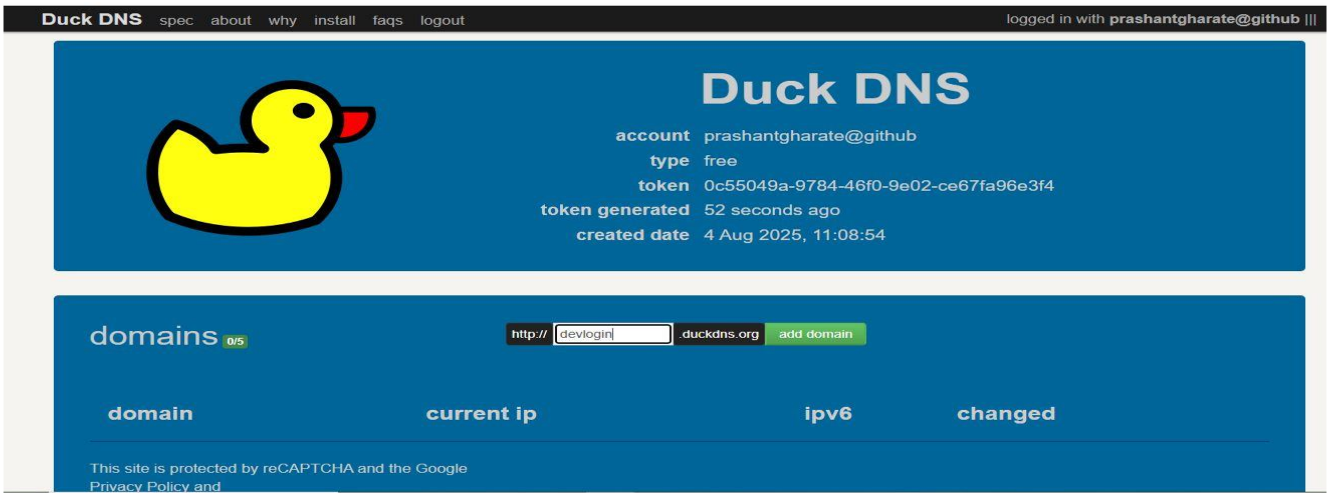

aws
Search [Alt+S]
Asia Pacific (Mumbai)
Prashant Gharate
EC2 > Security Groups > sg-0142864cb13cfe52f - launch-wizard-1 > Edit inbound rules
Add rule

Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-029f22b4cdf0e22a8	HTTPS	TCP	443	Cust...	0.0.0.0/0		Delete
sgr-03a2bea0133c804f7	HTTP	TCP	80	Cust...	0.0.0.0/0		Delete
sgr-01dd47002f9f54c32	SSH	TCP	22	Cust...	0.0.0.0/0		Delete
sgr-0c202ec287227938c	Custom TCP	TCP	8080	Cust...	0.0.0.0/0		Delete
–	Custom TCP	TCP	3000	Any...	0.0.0.0/0		Delete

Update your EC2 **Security Group** with these inbound rules:

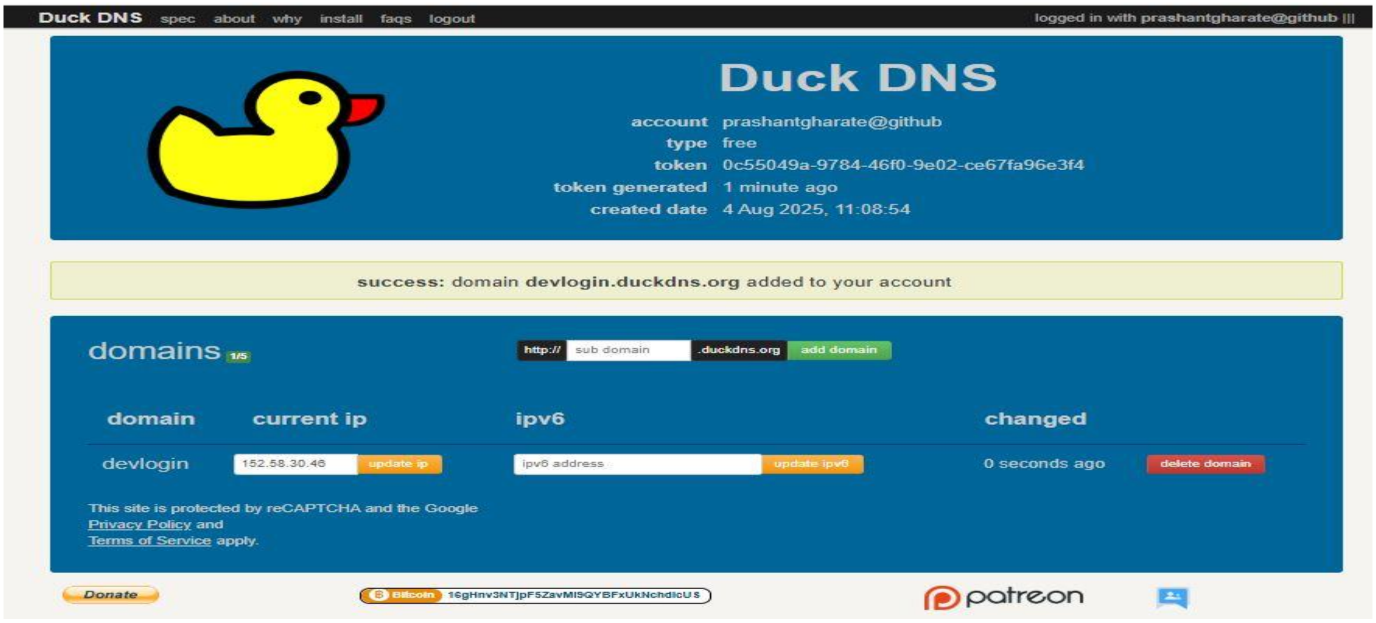
Step 8: Create Free Dynamic Domain using DuckDNS


Duck DNS spec about why install faqs logout
logged in with prashantgharate@github |||
Duck DNS
domains 0/5
http:// devlogin .duckdns.org add domain
domain current ip ipv6 changed
This site is protected by reCAPTCHA and the Google Privacy Policy and

account	prashantgharate@github
type	free
token	0c55049a-9784-46f0-9e02-ce67fa96e3f4
token generated	52 seconds ago
created date	4 Aug 2025, 11:08:54

- Go to: https://www.duckdns.org
- Log in with GitHub.
- Enter subdomain name (e.g., `devlogin`).
- Click **Add domain**.

Step 9: Domain Registered and Mapped



Duck DNS spec about why install faqs logout — logged in with prashantgharate@github |||

Duck DNS

account	prashantgharate@github
type	free
token	0c55049a-9784-46f0-9e02-ce67fa96e3f4
token generated	1 minute ago
created date	4 Aug 2025, 11:08:54

success: domain **devlogin.duckdns.org** added to your account

domains 1/5 — http:// sub domain .duckdns.org add domain

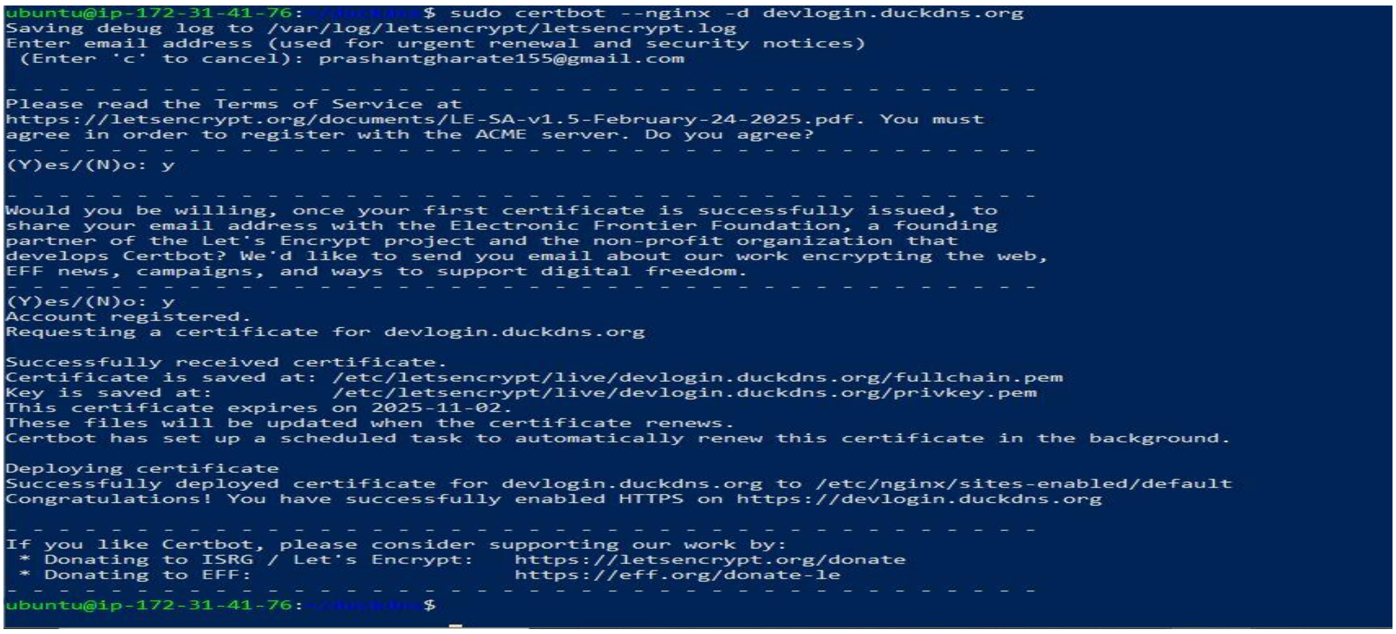
domain	current ip		ipv6		changed	
devlogin	152.58.30.46	update ip	ipv6 address	update ipv6	0 seconds ago	delete domain

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Donate — Bitcoin 16gHnv3NTjpF5ZavMI9QYBFxUkNchdicU $ — patreon

- Domain `devlogin.duckdns.org` is created.
- Current IP is automatically mapped to EC2's public IP.
- Token generated will be used for dynamic DNS updates (optional if IP is static).

Step 10: Install SSL Using Let's Encrypt (Certbot)



```
ubuntu@ip-172-31-41-76:~/duckdns$ sudo certbot --nginx -d devlogin.duckdns.org
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): prashantgharate155@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.5-February-24-2025.pdf. You must
agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y
Account registered.
Requesting a certificate for devlogin.duckdns.org

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/devlogin.duckdns.org/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/devlogin.duckdns.org/privkey.pem
This certificate expires on 2025-11-02.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

Deploying certificate
Successfully deployed certificate for devlogin.duckdns.org to /etc/nginx/sites-enabled/default
Congratulations! You have successfully enabled HTTPS on https://devlogin.duckdns.org

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
ubuntu@ip-172-31-41-76:~/duckdns$
```

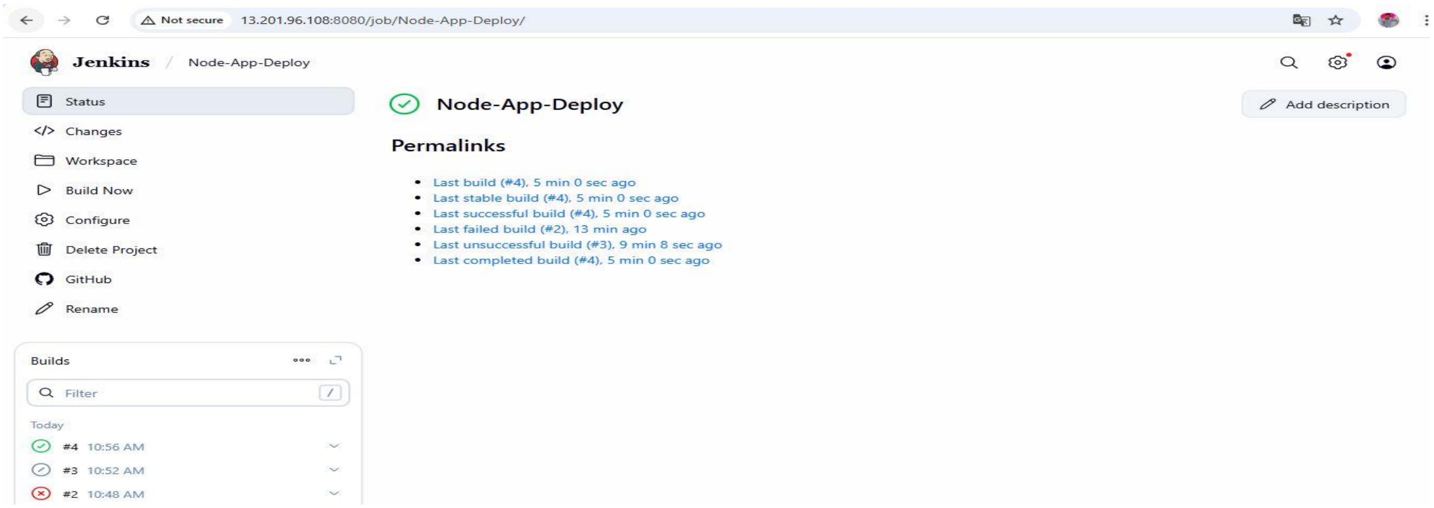
Install Certbot and secure the domain:

```
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d devlogin.duckdns.org
```

Follow the prompts:

- Enter your email.
- Agree to terms.
- Certbot will install and configure HTTPS automatically.
- Certificate is saved at `/etc/letsencrypt/live/devlogin.duckdns.org/`.

Step 11: Node.js App Deployment


13.201.96.108:8080/job/Node-App-Deploy/

Jenkins / Node-App-Deploy

Node-App-Deploy

Permalinks

- Last build (#4), 5 min 0 sec ago
- Last stable build (#4), 5 min 0 sec ago
- Last successful build (#4), 5 min 0 sec ago
- Last failed build (#2), 13 min ago
- Last unsuccessful build (#3), 9 min 8 sec ago
- Last completed build (#4), 5 min 0 sec ago

Builds

Today

#4 10:56 AM

#3 10:52 AM

#2 10:48 AM

We used Jenkins to automatically build and deploy our Node.js application to the EC2 instance. The build pipeline installed dependencies and started the app on port 3000.

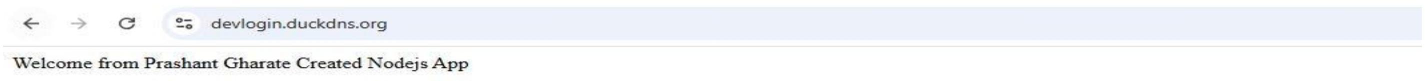
- Jenkins Job → Pull code from GitHub
- Execute `npm install`, `npm test` (optional), and start app using PM2

---

4. NGINX + HTTPS (Let's Encrypt)

- Installed NGINX: `sudo apt install nginx`
- Configured reverse proxy in `/etc/nginx/sites-available/default`:
  - Forwarded traffic from port 443 → 3000
- Installed Certbot & SSL cert: `sudo certbot --nginx -d devlogin.duckdns.org`
- Verified HTTPS access via: `https://devlogin.duckdns.org`

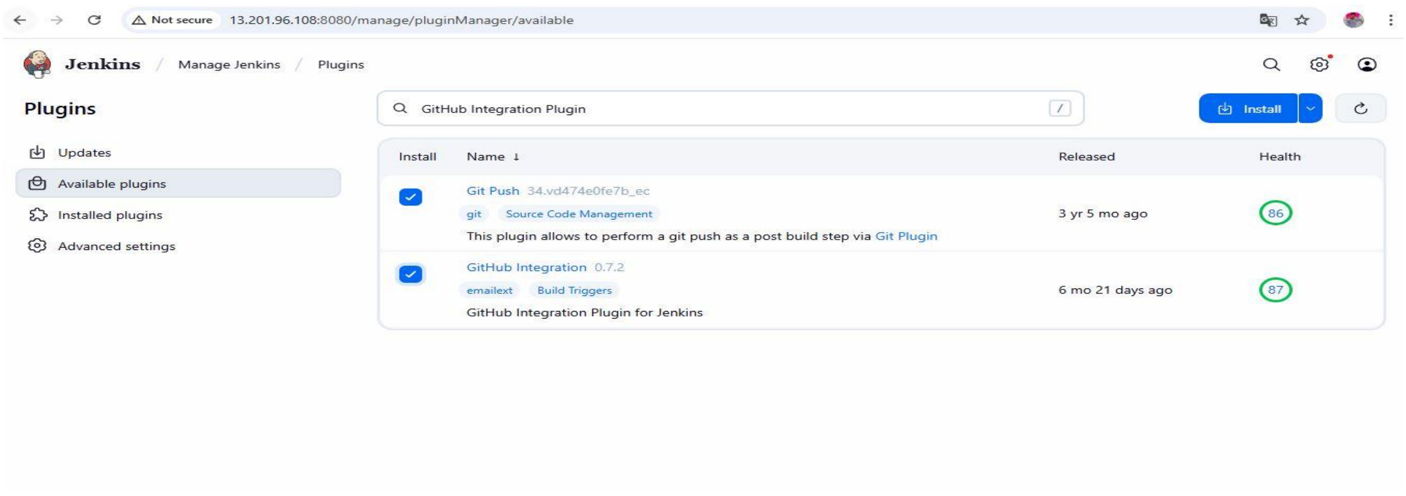
Step 12: Accessing the App via Custom Domain


devlogin.duckdns.org

Welcome from Prashant Gharate Created Nodejs App

The deployed Node.js app was accessed via a custom DuckDNS domain (e.g., https://devlogin.duckdns.org) with HTTPS secured using a Let's Encrypt certificate.

Step 13: Install GitHub Plugin in Jenkins


Not secure 13.201.96.108:8080/manage/pluginManager/available

Jenkins / Manage Jenkins / Plugins

Plugins

GitHub Integration Plugin

Install	Name ↓	Released	Health
✓	Git Push 34.vd474e0fe7b_ec git Source Code Management This plugin allows to perform a git push as a post build step via Git Plugin	3 yr 5 mo ago	86
✓	GitHub Integration 0.7.2 emailext Build Triggers GitHub Integration Plugin for Jenkins	6 mo 21 days ago	87

We installed the GitHub plugin in Jenkins:

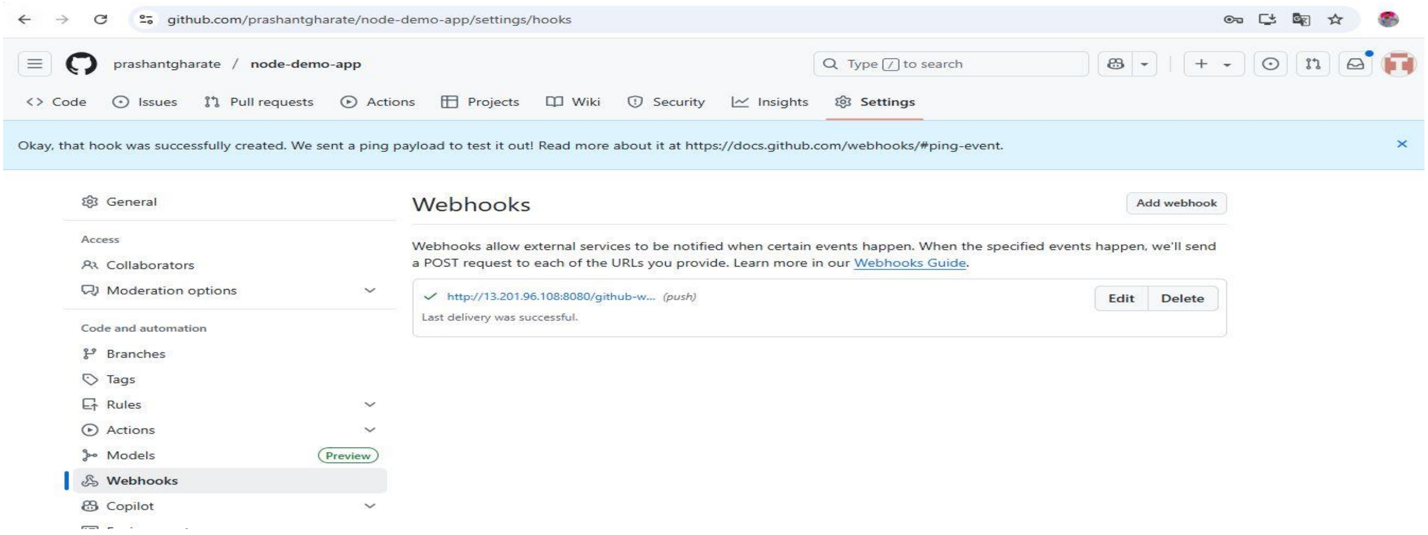
- Manage Jenkins → Plugin Manager → GitHub Plugin
  This enables integration with GitHub for Webhook-based automation.

---

5. GitHub Webhook Integration

- Created a GitHub repo for source code.
- Configured Webhook in GitHub → Settings → Webhooks:
  - Payload URL: `http://<jenkins-public-ip>:8080/github-webhook/`
  - Content type: `application/json`
- Installed Jenkins Plugins: GitHub Integration, GitHub Branch Source.

Step 14: Create GitHub Webhook


github.com/prashantgharate/node-demo-app/settings/hooks

prashantgharate / node-demo-app

Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://docs.github.com/webhooks/#ping-event.

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://13.201.96.108:8080/github-w... (push)
Last delivery was successful.

A webhook was configured in the GitHub repository to notify Jenkins whenever code is pushed.
URL format:
https://your-jenkins-domain/github-webhook/

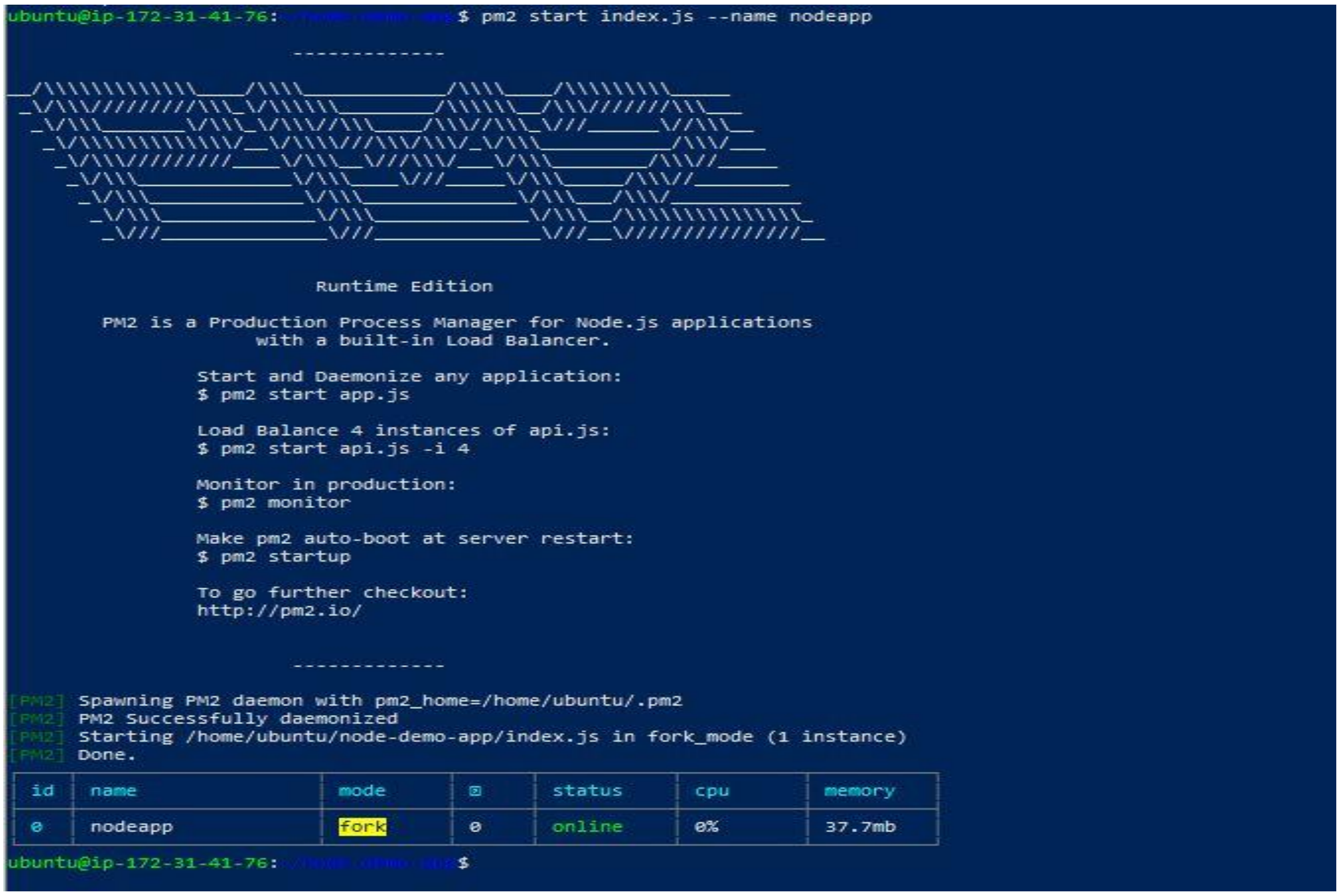
This triggers Jenkins automatically on push.

---

6. Jenkins Pipeline

- Created `Jenkinsfile` in root directory with following stages:
  - 🔧 Build: `npm install`
  - ? Test: `npm test` or `curl http://localhost:3000`
  - 🚀 Deploy: Executes `deploy.sh` to restart PM2 and reload NGINX.

Step 15: PM2 Monitoring Setup



```
ubuntu@ip-172-31-41-76:~/node-demo-app$ pm2 start index.js --name nodeapp

                        -------------

__/\\\\\\\\\\\\\____/\\\\____________/\\\\____/\\\\\\\\\_____
 _\/\\\/////////\\\_\/\\\\\\________/\\\\\\__/\\\///////\\\___
  _\/\\\_______\/\\\_\/\\\//\\\____/\\\//\\\_\///______\//\\\__
   _\/\\\\\\\\\\\\\/__\/\\\\///\\\/\\\/_\/\\\___________/\\\/___
    _\/\\\/////////____\/\\\__\///\\\/___\/\\\________/\\\//_____
     _\/\\\_____________\/\\\____\///_____\/\\\_____/\\\//________
      _\/\\\_____________\/\\\_____________\/\\\___/\\\/___________
       _\/\\\_____________\/\\\_____________\/\\\__/\\\\\\\\\\\\\\\_
        _\///______________\///______________\///__\///////////////__


                          Runtime Edition

        PM2 is a Production Process Manager for Node.js applications
                     with a built-in Load Balancer.

                Start and Daemonize any application:
                $ pm2 start app.js

                Load Balance 4 instances of api.js:
                $ pm2 start api.js -i 4

                Monitor in production:
                $ pm2 monitor

                Make pm2 auto-boot at server restart:
                $ pm2 startup

                To go further checkout:
                http://pm2.io/


                        -------------

[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ubuntu/node-demo-app/index.js in fork_mode (1 instance)
[PM2] Done.
│ id │ name                 │ mode     │ ↺    │ status    │ cpu      │ memory   │
│ 0  │ nodeapp              │ fork     │ 0    │ online    │ 0%       │ 37.7mb   │
ubuntu@ip-172-31-41-76:~/node-demo-app$
```

PM2 was used to run and monitor the Node.js application continuously.

Commands used:

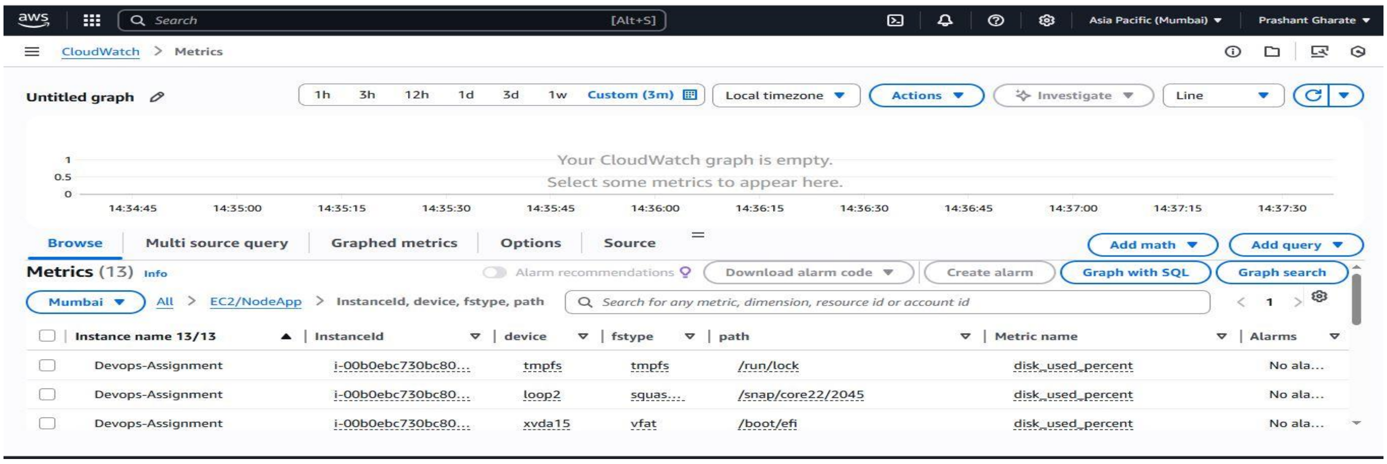
```
sudo npm install -g pm2
pm2 start index.js --name nodeapp
pm2 save
pm2 startup
```

Use `pm2 monit` or `pm2 logs` to monitor the running app.

Step 16: Install and Configure CloudWatch Agent

7. AWS CloudWatch Monitoring

- Installed CloudWatch agent: `sudo apt install amazon-cloudwatch-agent`
- Created config file at: `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`
- Started agent: `amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:<path> -s`
- Verified metrics in CloudWatch Console.



Metrics (13) Info

All > EC2/NodeApp > InstanceId, device, fstype, path

Instance name 13/13	InstanceId	device	fstype	path	Metric name	Alarms
Devops-Assignment	i-00b0ebc730bc80...	tmpfs	tmpfs	/run/lock	disk_used_percent	No ala...
Devops-Assignment	i-00b0ebc730bc80...	loop2	squas...	/snap/core22/2045	disk_used_percent	No ala...
Devops-Assignment	i-00b0ebc730bc80...	xvda15	vfat	/boot/efi	disk_used_percent	No ala...

We installed the Amazon CloudWatch Agent on our EC2 instance to collect metrics like:

- CPU
- Memory
- Disk

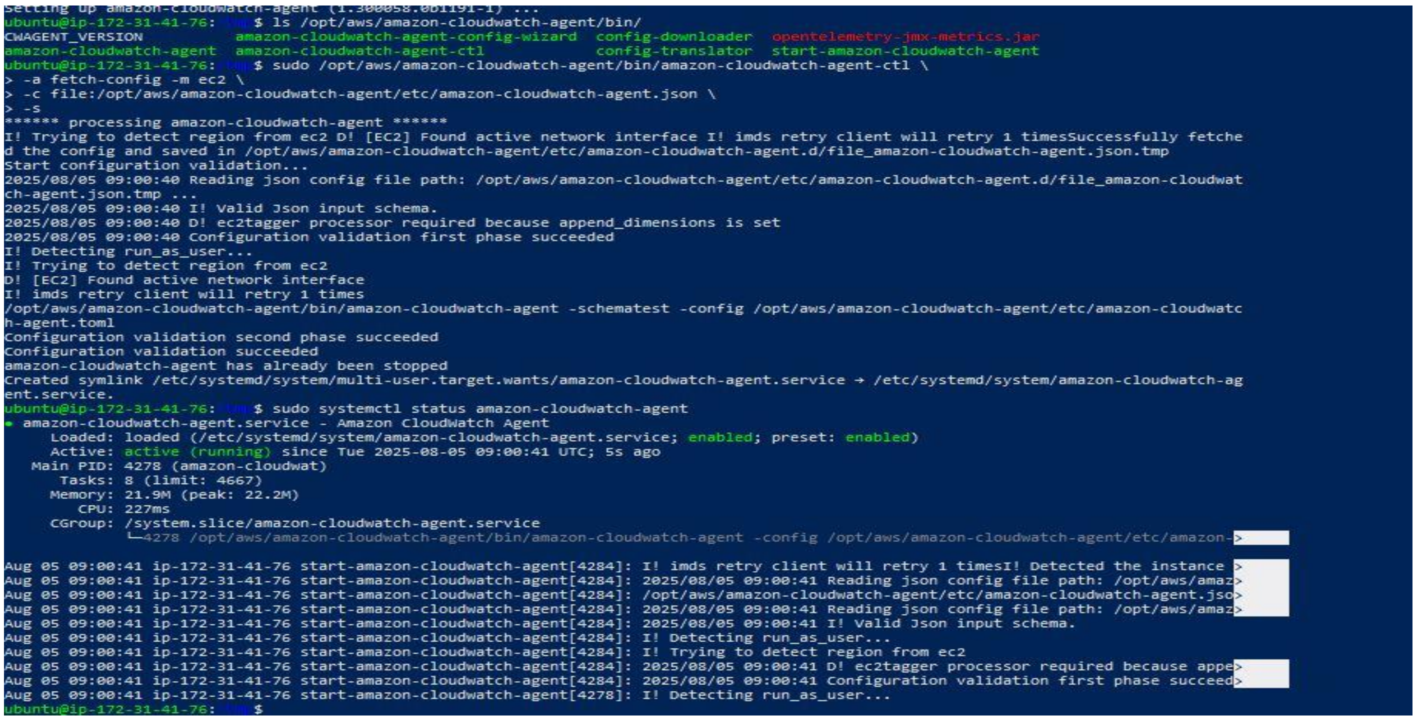
Steps:

```
sudo yum install amazon-cloudwatch-agent
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
  -a fetch-config -m ec2 \
  -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json \
  -s
```

Step 17: View Metrics in CloudWatch

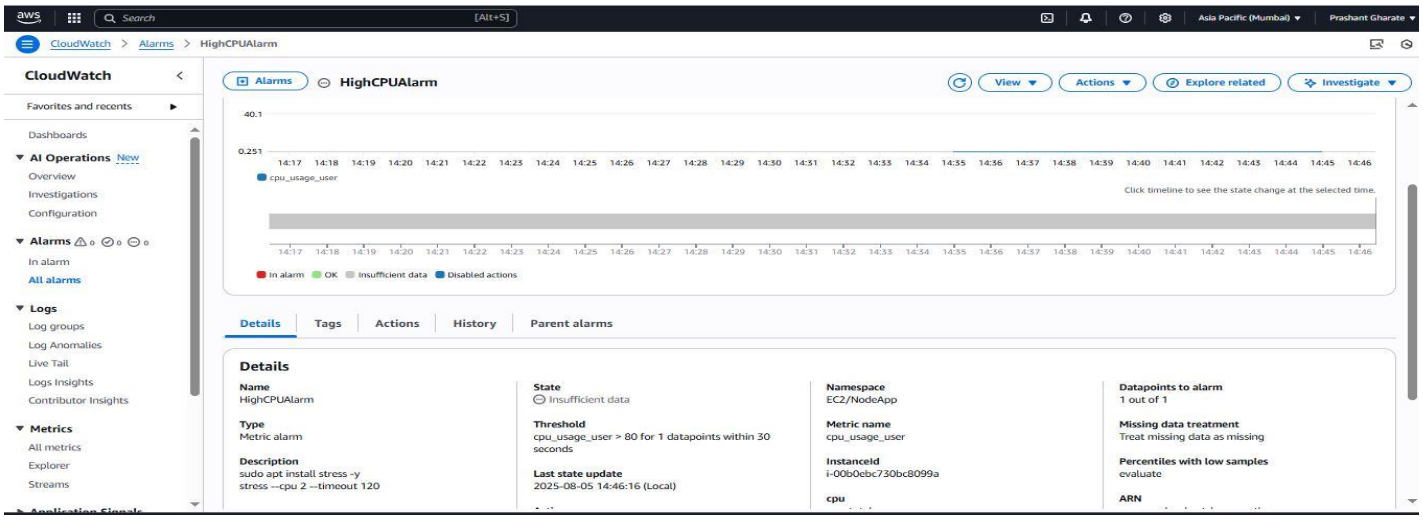
8. CloudWatch Alarm + Email

- Go to AWS CloudWatch → Alarms → Create Alarm.
- Metric: CPU Utilization or Custom log metric.
- Set threshold (e.g., > 70% CPU).
- Create SNS topic and add your email as a subscriber.
- Confirm email and link to alarm.



```
Setting up amazon-cloudwatch-agent (1.300058.0b1191-1) ...
ubuntu@ip-172-31-41-76:/tmp$ ls /opt/aws/amazon-cloudwatch-agent/bin/
CWAGENT_VERSION          amazon-cloudwatch-agent-config-wizard  config-downloader  opentelemetry-jmx-metrics.jar
amazon-cloudwatch-agent  amazon-cloudwatch-agent-ctl            config-translator  start-amazon-cloudwatch-agent
ubuntu@ip-172-31-41-76:/tmp$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
> -a fetch-config -m ec2 \
> -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json \
> -s
****** processing amazon-cloudwatch-agent ******
I! Trying to detect region from ec2 D! [EC2] Found active network interface I! imds retry client will retry 1 timesSuccessfully fetche
d the config and saved in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_amazon-cloudwatch-agent.json.tmp
Start configuration validation...
2025/08/05 09:00:40 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_amazon-cloudwat
ch-agent.json.tmp ...
2025/08/05 09:00:40 I! Valid Json input schema.
2025/08/05 09:00:40 D! ec2tagger processor required because append_dimensions is set
2025/08/05 09:00:40 Configuration validation first phase succeeded
I! Detecting run_as_user...
I! Trying to detect region from ec2
D! [EC2] Found active network interface
I! imds retry client will retry 1 times
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatc
h-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
amazon-cloudwatch-agent has already been stopped
Created symlink /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service → /etc/systemd/system/amazon-cloudwatch-ag
ent.service.
ubuntu@ip-172-31-41-76:/tmp$ sudo systemctl status amazon-cloudwatch-agent
● amazon-cloudwatch-agent.service - Amazon CloudWatch Agent
     Loaded: loaded (/etc/systemd/system/amazon-cloudwatch-agent.service; enabled; preset: enabled)
     Active: active (running) since Tue 2025-08-05 09:00:41 UTC; 5s ago
   Main PID: 4278 (amazon-cloudwat)
      Tasks: 8 (limit: 4667)
     Memory: 21.9M (peak: 22.2M)
        CPU: 227ms
     CGroup: /system.slice/amazon-cloudwatch-agent.service
             └─4278 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-agent/etc/amazon->

Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: I! imds retry client will retry 1 timesI! Detected the instance >
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: 2025/08/05 09:00:41 Reading json config file path: /opt/aws/amaz>
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.jso>
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: 2025/08/05 09:00:41 Reading json config file path: /opt/aws/amaz>
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: 2025/08/05 09:00:41 I! Valid Json input schema.
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: I! Detecting run_as_user...
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: I! Trying to detect region from ec2
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: 2025/08/05 09:00:41 D! ec2tagger processor required because appe>
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4284]: 2025/08/05 09:00:41 Configuration validation first phase succeed>
Aug 05 09:00:41 ip-172-31-41-76 start-amazon-cloudwatch-agent[4278]: I! Detecting run_as_user...
ubuntu@ip-172-31-41-76:/tmp$
```



Details

Name
HighCPUAlarm

Type
Metric alarm

Description
sudo apt install stress -y
stress --cpu 2 --timeout 120

State
Insufficient data

Threshold
cpu_usage_user > 80 for 1 datapoints within 30 seconds

Last state update
2025-08-05 14:46:16 (Local)

Namespace
EC2/NodeApp

Metric name
cpu_usage_user

InstanceId
i-00b0ebc730bc8099a

cpu

Datapoints to alarm
1 out of 1

Missing data treatment
Treat missing data as missing

Percentiles with low samples
evaluate

ARN

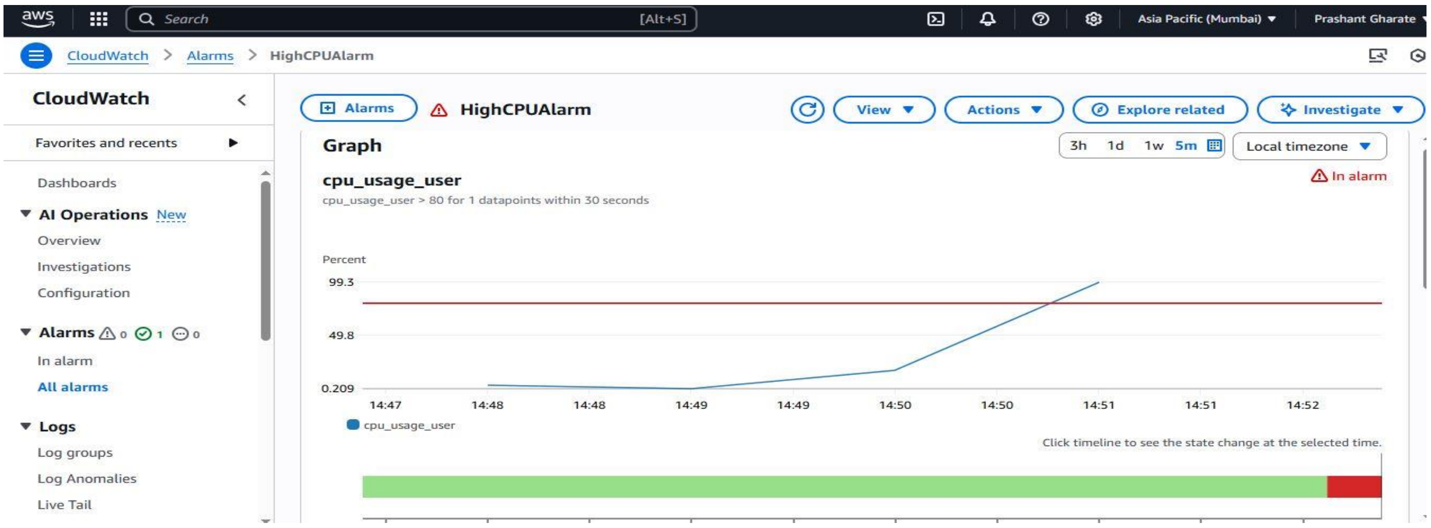
Once the CloudWatch Agent is running, we navigated to AWS → CloudWatch → Metrics → EC2 → Instance Metrics to view:

- CPUUtilization
- MemoryUsed
- DiskSpaceUtilization
- etc.

Step 18: Create CloudWatch Alarm

9. Final Testing

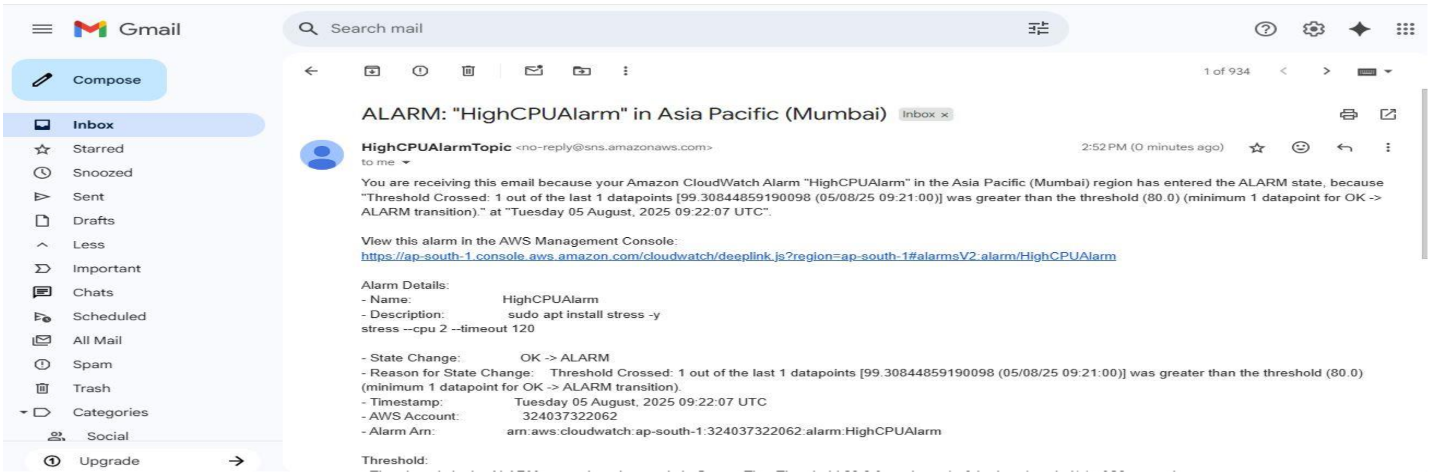
- Local test: `curl http://localhost:3000` → Hello Prashant!
- Public test: `https://devlogin.duckdns.org` → Works with SSL
- Monitoring: `pm2 monit` and AWS CloudWatch Dashboard


CloudWatch > Alarms > HighCPUAlarm
HighCPUAlarm
Graph
cpu_usage_user
cpu_usage_user > 80 for 1 datapoints within 30 seconds
In alarm
Percent
99.3
49.8
0.209
14:47
14:48
14:48
14:49
14:49
14:50
14:50
14:51
14:51
14:52
cpu_usage_user
Click timeline to see the state change at the selected time.

We set up a CloudWatch Alarm to monitor high CPU usage.
Example:

- If CPU > 70% for 2 periods of 5 minutes → trigger alarm.

Step 19: Email Notification via SNS



ALARM: "HighCPUAlarm" in Asia Pacific (Mumbai)

HighCPUAlarmTopic <no-reply@sns.amazonaws.com>
to me
2:52 PM (0 minutes ago)

You are receiving this email because your Amazon CloudWatch Alarm "HighCPUAlarm" in the Asia Pacific (Mumbai) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [99.30844859190098 (05/08/25 09:21:00)] was greater than the threshold (80.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Tuesday 05 August, 2025 09:22:07 UTC".

View this alarm in the AWS Management Console:
https://ap-south-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=ap-south-1#alarmsV2:alarm/HighCPUAlarm

```
Alarm Details:
- Name:                    HighCPUAlarm
- Description:             sudo apt install stress -y
stress --cpu 2 --timeout 120

- State Change:            OK -> ALARM
- Reason for State Change:    Threshold Crossed: 1 out of the last 1 datapoints [99.30844859190098 (05/08/25 09:21:00)] was greater than the threshold (80.0)
(minimum 1 datapoint for OK -> ALARM transition).
- Timestamp:               Tuesday 05 August, 2025 09:22:07 UTC
- AWS Account:             324037322062
- Alarm Arn:               arn:aws:cloudwatch:ap-south-1:324037322062:alarm:HighCPUAlarm

Threshold:
```

To get alerted, we created:

1. An SNS Topic
2. An Email Subscription to that Topic
3. Linked the SNS Topic to the CloudWatch Alarm

Now whenever the alarm state goes to "ALARM", we get an email instantly